Sharding in MongoDB is a data partitioning technique used to distribute data across multiple servers or clusters to achieve horizontal scalability. It allows MongoDB to handle large amounts of data and high write and read throughput by distributing the data across multiple machines. Sharding is one of the key features that enable MongoDB to scale out and support large-scale applications.

Here's how sharding works in MongoDB:

1. **Shard Key:**
    - The first step in sharding is to choose a shard key. The shard key is a field or set of fields in the documents of a collection. MongoDB uses this key to determine how data is distributed across shards.
    - The shard key should be carefully chosen based on the access patterns of your application and the type of queries that will be performed. An ideal shard key evenly distributes data across shards to prevent hotspots and unbalanced data distribution.
2. **Shard Cluster:**
    - A MongoDB sharded cluster consists of three main components: **config servers**, **shard servers**, and **mongos** instances.
    - Config Servers: These servers store the metadata of the cluster, including the mapping of shard keys to chunks (ranges of the shard key). Typically, there are three config servers for redundancy.
    - Shard Servers: These servers store the actual data. Data is divided into chunks, and each chunk is stored on a different shard server.
    - Mongos Instances: These are routing processes that sit between the application and the cluster. They receive queries from the application, determine the relevant shards to access based on the shard key, and forward the queries to the appropriate shard servers.
3. **Data Distribution:**
    - When data is inserted or added to the sharded collection, MongoDB hashes the shard key and determines which shard the data belongs to.
    - The data is then split into chunks based on the shard key range and distributed across the shards. MongoDB automatically balances the chunks to ensure an even distribution of data.
    - As the data size and distribution change, MongoDB may need to move chunks between shards to maintain a balanced cluster.
4. **Query Routing:**

- When a query is sent to the sharded cluster via a **mongos** instance, the **mongos** routes the query to the appropriate shard servers based on the shard key in the query predicate.
- By knowing the shard key, **mongos** can quickly determine the shard(s) that should have the relevant data, thus allowing the query to be executed efficiently.

5. **Scaling and Adding Shards:**
    - As the data and workload increase, you can scale the sharded cluster by adding more shards to the system. This allows MongoDB to distribute data across a larger number of servers, increasing the capacity and performance of the database.

Sharding in MongoDB provides horizontal scaling, allowing you to handle growing data and traffic by distributing the data across multiple servers. It is essential to design the shard key carefully and monitor the cluster's performance to ensure an optimal and balanced sharding setup.